Introduction to Databases and SQL

ЛЕКЦИЯ 3

Темы занятия

Разбор задания №1

Изменение структуры таблицы

Удаление таблиц и баз

Первичный ключ

Связи между таблицами

Изменение структуры таблицы

Для изменения структуры таблицы служит инструкция `ALTER TABLE`. Можно делать следующие изменения:

- добавлять и удалять колонки;
- изменять свойства колонок;
- добавлять и удалять именованные ограничения;

Внимание: следует учитывать нюансы при изменении структуры непустой таблицы.

Добавление колонки

Чтобы добавить новую колонку, в `ALTER TABLE` используется предложение `ADD`:

```
ALTER TABLE Employee
ADD PhoneNumber char(12) NULL
```

В одной инструкции `ALTER TABLE` можно добавить только одну колонку.

Удаление колонки

Колонки из таблицы удаляются при помощи предложения `DROP COLUMN`:

```
ALTER TABLE Employee
DROP COLUMN PhoneNumber
```

Изменение свойств колонки

Для изменения свойств существующей колонки применяется предложение `ALTER COLUMN`. Модификации поддаются следующие свойства колонки:

- тип данных;
- свойство колонки хранить значения `NULL`.

```
ALTER TABLE Employee
ALTER COLUMN FirstName char(25) NOT NULL
```

Добавление и удаление ограничений

Для добавления в таблицу именованного ограничения используется предложение `ADD CONSTRAINT`:

```
ALTER TABLE Employee
ADD CONSTRAINT uc_FirstName UNIQUE (FirstName)
```

Удалить именованное ограничение можно при помощи предложения `DROP CONSTRAINT`:

```
ALTER TABLE Employee
DROP CONSTRAINT uc_FirstName
```

Удаление таблицы

Инструкция `DROP TABLE` служит для удаления таблиц (ы):

```
-- удаляем одну таблицу
DROP TABLE Profiles

-- удаляем сразу три таблицы
DROP TABLE Roles, Users, UserRoles
```

Удаление базы

Инструкция `DROP DATABASE` служит для безвозвратного удаления одной или нескольких баз:

```
-- удаляем одну базу (несколько – через запятую)
DROP DATABASE Projects
```

Первичный ключ

Первичный ключ (primary key) – колонка (или набор колонок) с уникальными значениям, позволяющими однозначно идентифицировать строки таблицы.

Каждая таблица может содержать только один первичный ключ (хотя самих колонок с уникальными значениями в таблице может быть несколько – это *потенциальные ключи*).

Выбор первичного ключа

1. Обычно первичный ключ выбирают на этапе проектирования таблицы, когда самих данных ещё нет. Надо **прогнозировать**, какие данные могут появиться в таблице (например, всегда ли комбинация из имени человека и его даты рождения будет уникальной).

2. Из нескольких потенциальных ключей первичным назначают обычно самый компактный.

Разновидности первичного ключа (ПК)

Атомарный (*простой*) ПК – состоит из одной колонки.

Составной ПК – состоит из нескольких колонок.

Естественный ПК – строится на уже существующих атрибутах сущности.

Суррогатный ПК – состоит из специально добавленных к сущности атрибутах (с уникальными значениями).

Задание первичного ключа – способ 1

```
CREATE TABLE Employee
(
  EmployeeID int PRIMARY KEY,
  FirstName nvarchar(50),
  LastName nvarchar(50),
  DepartmentID char(4)
)
```

Только одну колонку можно пометить как `PRIMARY KEY`. Она автоматически будет `NOT NULL`.

Задание первичного ключа – способ 2

```
CREATE TABLE Employee
(
  EmployeeID int NOT NULL,
  FirstName nvarchar(50),
  LastName nvarchar(50),
  DepartmentID char(4),
  CONSTRAINT pk_emp PRIMARY KEY(EmployeeID)
)
```

Так можно построить ключ по нескольким колонкам!

Колонка идентификаторов

В T-SQL **одну целочисленную** колонку таблицы можно сделать *колонкой идентификаторов*.

При добавлении строки в таблицу значения в такой колонке будут формироваться **автоматически**: это будут элементы числовой последовательности с заданным начальным значением и шагом.

Колонка идентификаторов – пример 1

```
CREATE TABLE Employee
(
  EmployeeID int PRIMARY KEY IDENTITY,
  FirstName nvarchar(50),
  LastName nvarchar(50),
  DepartmentID char(4)
)
```

`EmployeeID` будет равен `1`, `2`, `3`, …

Колонка идентификаторов – пример 2

```
CREATE TABLE Employee
(
  EmployeeID int PRIMARY KEY IDENTITY(5, 3),
  FirstName nvarchar(50),
  LastName nvarchar(50),
  DepartmentID char(4)
)
```

`EmployeeID` будет равен `5`, `8`, `11`, …

Связи между таблицами

Пусть в таблице T1 есть первичный (или потенциальный) ключ PK. Пусть в таблице T2 колонка (набор колонок) FK принимает значения из множества значений PK.

В этом случае будем говорить о том, что таблицы T1 и T2 *связаны по ключу PK*. Колонка (или колонки) FK называется в таблице T2 *внешним ключом* (foreign key).

T1 – *главная таблица*, T2 – *зависимая таблица*.

Связи между таблицами – пример

UserID	Login	Password
10	alexv	qwerty
20	ivan	123
30	dasha93	password
40	oleg	&80__12r

RoleID	RoleName
1	Admin
2	Editor
3	User

Таблица `Users` хранит данные пользователей, таблица `Roles` (первичный ключ `RoleID`) описывает роли. Надо связать таблицы, чтобы у каждого пользователя была ровно одна роль.

Связи между таблицами – пример

UserID	Login	Password	UserRole
10	alexv	qwerty	2
20	ivan	123	3
30	dasha93	password	3
40	oleg	&80__12r	1

RoleID	RoleName
1	Admin
2	Editor
3	User

Колонка `UserRole` – внешний ключ в таблице `Users`.

Создание связи – 1

Во-первых, нужна та таблица, **на которую** будем ссылаться (*главная таблица*):

```
CREATE TABLE Roles
(
  RoleID int PRIMARY KEY,
  RoleName nvarchar(50) NOT NULL
)
```

Создание связи – 2

Во-вторых, нужна таблица, **которая** будет ссылаться.

```
CREATE TABLE Users
(
  UserID int PRIMARY KEY,
  [Login] nvarchar(50) NOT NULL,
  [Password] nvarchar(50) NOT NULL,
  UserRole int NOT NULL,
  CONSTRAINT fk_roles FOREIGN KEY (UserRole)
                      REFERENCES Roles(RoleID)
)
```

Назначение связей между таблицами

Связь обеспечивает *ссылочную целостность* данных.

При наличии связи нельзя вставить в зависимую таблицу строку со значениями внешнего ключа, отсутствующими в главной таблице.

Типы связей между таблицами

Пусть T2 – таблица с внешним ключом на таблицу T1.

1. Связь «*один ко многим*». Каждой строке из T1 соответствует несколько строк из T2 (0, 1, .. N строк).

2. Связь «*один к одному*». Каждой строке из T1 соответствует одна строка из T2 (или ноль строк).

3. Связь «*многие ко многим*». Каждой строке из T1 соответствует несколько строк из T2, и **наоборот**.

Связь «один ко многим»

UserID	Login	Password	UserRole
10	alexv	qwerty	2
20	ivan	123	3
30	dasha93	password	3
40	oleg	&80__12r	1

RoleID	RoleName
1	Admin
2	Editor
3	User

Это самый распространённый тип связи.

Связь «один к одному»

EmployeeID	Salary
1	1000
3	2000

PersonID	Name
1	Alex
2	John
3	Mary

Такая связь описывает отношение *уточнения* или *наследования*.

Связь «многие ко многим»

ID	BookTitle
1	Сияние
2	Талисман
3	Чёрный дом

ID	AuthorName
1	Стивен Кинг
2	Питер Страуб

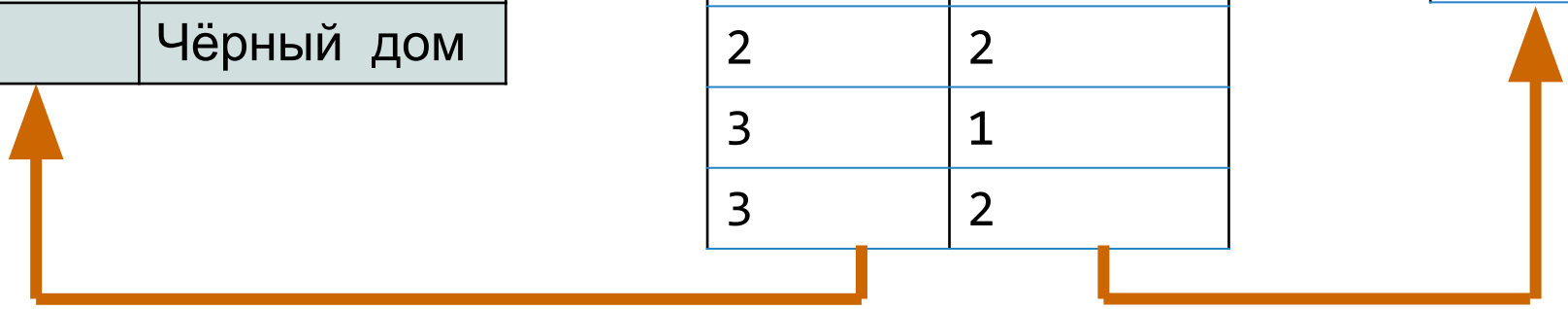
Один автор может написать несколько книг, но и у книги может быть несколько авторов.

Связь «многие ко многим»

ID	BookTitle
1	Сияние
2	Талисман
3	Чёрный дом

BookID	AuthorID
1	1
2	1
2	2
3	1
3	2

ID	AuthorName
1	Стивен Кинг
2	Питер Страуб



Этот *логический* тип связи на практике реализуется при помощи **дополнительной таблицы**.

Диаграмма базы данных

Диаграмма БД позволяет наглядно представить структуру таблиц и связей между ними (*схема БД*).

Многие СУБД содержат средства для построения диаграммы по выбранным таблицам базы.

Некоторые СУБД позволяют строить диаграмму, а затем на её основе сгенерировать схему базы данных.

Диаграмма базы данных – пример